Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga

TEMA 4: Estructuras de Control

Fundamentos de Informática
(Ingeniería Técnica Industrial)

E. U. Politécnica

Autores: M.C.Aranda,A.Fernández,J.Galindo,M.Trella

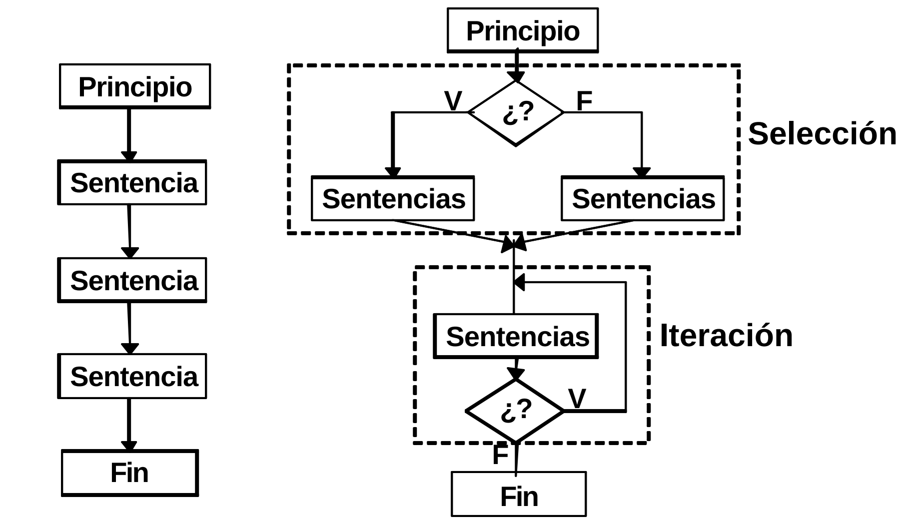
Índice de contenidos

Introducción: 3 estructuras de control

- **Estructura secuencial**: aquella en la que las instrucciones o sentencias son ejecutadas una a una en orden.

- Se puede alterar esa secuencialidad usando dos **estructuras o sentencias de control** distintas. Estas estructuras permiten variar el flujo de control del programa dependiendo de ciertas condiciones. Son:
  - **Estructura de selección**: Permite que se tomen rutas alternativas de acción dependiendo del resultado de una operación.
  - **Estructura repetitiva (iteración o bucle)**: Permite repetir un conjunto de sentencias.

Introducción: 3 estructuras de control



Estructura secuencial

Estructuras no secuenciales

1.1. Sentencias `if, if-else`

- Sirven para elegir entre una acción u otra dependiendo de una **condición**.

```
if (condición) {
    Bloque Verdad
}
```

```
if (condición) {
     Bloque Verdad
}
else {
     Bloque Falso
}
```

- Una **condición** es una expresión que puede ser verdadera o falsa.
  - Recordemos que en C no existen tipos lógicos, el valor verdadero es cualquier valor distinto de cero y falso es el valor 0 .
  - De hecho, más que una condición puede haber cualquier **expresión**.
- La condición SIEMPRE debe ir entre paréntesis.
- **Funcionamiento**: Se determina si la condición es cierta o falsa. Si es cierta, el programa ejecuta un bloque de código (el Bloque Verdad) y si es falsa se ejecuta otro bloque distinto si lo hay (el Bloque Falso).
- Si el bloque sólo tiene una sentencia, las llaves se pueden quitar.

1.1. Ejemplo: Expresiones

- Dada la siguiente declaración de variables:
  `int a=5, b=10;`
- Decir el valor numérico y el valor lógico (Verdad o Falso) de las siguientesexpresiones:

Expresión	Valor numérico	Valor lógico
a	5	V
!a	0	F
!!a	1	V
a-b	-5	V
a-b+5	0	F
!(a-b+5)	1	V
a<b	1	V
!(!a || !b)	1	V
!(a && b)	0	F

<u>Nota</u>: Cualquier expresión es válida como condición en un `if` o `if-else`.

1.1. Ejemplo: Sentencias `if, if-else`

```
/* Determina el número más grande entre tres */
#include <stdio.h>
void main() {
  int n1,n2,n3,mayor;
  printf("Primer valor: ");scanf("%d",&n1);
  printf("Segundo valor: ");scanf("%d",&n2);
  printf("Tercer valor: ");scanf("%d",&n3);
  if(n1<n2) /* Calcular el mayor de n1 y n2 */
    mayor=n2;
  else
    mayor=n1;
  if(mayor<n3) /* Ver si n3 es el mayor */
    mayor=n3;
  printf("El mayor de (%d,%d,%d) es %d\n",
              n1,n2,n3,mayor);
}
```


1. Ejemplo: Sentencia `if-else`

```
/* Calcular el flujo de corriente en un circuito
eléctrico de resistencia r al que se le aplica un
voltaje de v. Se calcula con la ecuación i=v/r.
Para evitar la posibilidad de dividir por cero
habrá que evaluar la expresión sólo si r no es
cero.*/
#include <stdio.h>
void main() {
  float r,v;
  printf("Resistencia? "); scanf("%f", &r);
  printf("Voltaje? "); scanf("%f", &v);
  if(r==0)
    printf("Existe un cortocircuito\n");
  else
    printf("Corriente = %f amps\n", v/r);
}
```

1.1. Ejemplo: Sentencias `if` anidadas

```
/* Determina si un número es menor, mayor o
igual a cero */
#include <stdio.h>
void main() {
  int i;
  printf("Valor del número: ");
  scanf("%d", &i);
  if (i>=0)
    if(i==0)
      printf("Número igual a cero\n");
    else
      printf("Número mayor que cero\n");
  else
    printf("Número menor que cero\n");
}
```


1.1. Sentencia `if-else` anidada

```
if (condición1) {
     Bloque1
}
else if (condición2){
     Bloque2
}
else if (condición3){
     Bloque3
}
else {
     Bloque4
}
```

- **Funcionamiento:**
  - El Bloque1 se ejecuta si la condición1 es cierta. Si es falsa el programa evalúa la siguiente condición.
  - Si la condición2 es cierta el programa ejecuta el Bloque2, en caso contrario evalúa la siguiente condición y así sucesivamente.
  - Si todas las condiciones son falsas el programa ejecuta el bloque de sentencias del `else` final (Bloque 4).
  - Por supuesto, este último `else` puede no existir.

1.1 Ejemplo: `if-else` anidados

```
/* Programa que devuelve la calificación obtenida
según la nota numérica*/
#include <stdio.h>
void main() {
   float nota;
   printf("Dame la nota: ");
   scanf("%f", &nota);
   if (nota==10)      puts("Matrícula de Honor");
   else if (nota>=9) puts("Sobresaliente");
   else if (nota>=7) puts("Notable");
   else if (nota>=5) puts("Aprobado");
   else puts("Suspenso");
}
```

Ejercicio propuesto: Modifica el programa anterior para que muestre mensajes de error si la nota leída es mayor que 10 o menor que cero.

1.1 Ejemplo: `if-else` anidados

```
/* Programa que clasifica una onda electromagnética (EM)
dada su longitud de onda */
#include <stdio.h>
void main() {
   float lambda;
   printf("Dame la longitud de onda: ");
   scanf("%f", &lambda);
   printf("La onda electromagnética es ");
   if   (lambda<1e-11) puts("Rayos Gamma !!!");
   else if  (lambda<1e-9)  puts("Rayos X !!!");
   else if  (lambda<400e-9)  puts("Ultravioleta !!!");
   else if  (lambda<700e-9)  puts("Luz !!!");
   else if  (lambda<1e-3)  puts("Infrarrojos !!!");
   else if  (lambda<1e-1)  puts("Microondas !!!");
   else puts("Ondas de radio !!!");
}
```

Ejercicio propuesto: Las ondas EM también se pueden especificar por su frecuencia. Modifica el programa anterior para que permita entrar la frecuencia de la onda y determine la longitud de onda usando la fórmula: $\lambda=c/f$, donde c es la velocidad de la luz (3e8) y f la frecuencia de la onda.

1.1. Ejemplo: `if-else` anidados

```
/* Determina la resistencia equivalente de dos resistencias
conectadas en serie o en paralelo */
#include <stdio.h>
#include <ctype.h> /* para la función tolower() */
void main() {
  float R1,R2,Requ;
  char ch;
  printf("Valor de la primera resistencia: "); scanf("%f",&R1);
  printf("Valor de la segunda resistencia: "); scanf("%f",&R2);
  fflush(stdin); /* Limpia la entrada estándar */
  printf("Las resistencias están en (s)erie o en (p)aralelo?");
  ch=tolower(getchar());/* lee un carácter y lo pasa a minúscula */
  if(ch=='s') {
    Requ=R1+R2;
    printf("Resistencia equivalente en serie es %f ohms\n", Requ);
  }
  else if(ch=='p'){
    Requ= R1*R2 /(R1+R2);
    printf("Resistencia equivalente en paralelo es %f ohms\n", Requ);
  }
  else puts("Entrada no válida.");
}
```


1.1. Ejemplo: Sentencia `if`

```
/* Raíces de una ecuación de
segundo grado */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void main() {
  double a,b,c,div,dis,r1;
  printf("Primer coeficiente: ");
  scanf("%f",&a);
  printf("Segundo coeficiente: ");
  scanf("%f",&b);
  printf("Tercer coeficiente: ");
  scanf("%f",&c);
  /* Si a vale 0, la ecuación se
  reduce a  bx+c=0 y x=-c/b,
  si b no es igual a cero.
  Si c y a son 0, entonces x=0 */
  if(a==0)
    if(c==0)
      printf("La raíz vale 0\n");
    else
      printf("Sólo hay una \
solución para x = %g\n", -c/b);
  /* Si a no vale 0 y b*b >=4*a*c,
  hay dos valores reales para x.
  Si b*b<4*a*c, la solución es
  compleja. */
  else { /* si a no es igual a 0 */
     div = 2.0*a;
     dis = b*b-4.0*a* c;
     if(dis>=0) {
      printf("Raíces reales\n");
      printf("Raíz 1 = %g\n",
        (-b +  sqrt(dis))/div);
      printf("Raíz 2 = %g\n",
        (-b -  sqrt(dis))/div);
     else { /* Raíces complejas */
      div= 2.0 * a;
      r1 = sqrt(fabs(dis))/div;
      printf("Raíces complejas\n");
      printf("Raíz 1 = (%g) + \
             i(%g)\n", -b/div, r1);
      printf("Raíz 2 = (%g) - \
             i(%g)\n", -b/div, r1);
     }
  }
}
```


1.2. Operador condicional `?:`

- Es una herramienta útil para evaluar expresiones condicionales.
- Su forma general es la siguiente:

  `expresión1?expresión2:expresión3;`

- Interpretación:
  - Si la `expresión1` es cierta, entonces se evalúa la `expresión2`, en otro caso se evalúa la `expresión3`.
  - Por tanto, el resultado de la expresión completa es el valor y tipo de la expresión evaluada: de la `expresión2` o de la `expresión3` (los cuales deben ser del mismo tipo).
- Ejemplo: `a=`**`b<0?-b:b`**`;`
  - La expresión condicional es la que está en negrita. Si el valor de `'b'` es menor que 0, la expresión completa tomará el valor de `'-b'`, en otro caso tomará el valor de `'b'`.
  - En definitiva, a la variable `'a'` se le asigna el valor absoluto de `'b'` dependiendo de la condición `'b<0'`. La sentencia anterior completa es equivalente a:

```
if(b<0)  a=-b;
else a=b;
```


1.3. Sentencia `switch`

```
switch (<expresión>) {
   case <valor1>: Bloque 1
                  break;
   case <valor2>: Bloque 2
                  break;
   case <valor3>: Bloque 3
                  break;
   default: Bloque 4;
}
```

- La sentencia **`switch`** se usa cuando existe una decisión múltiple.
- Normalmente se usa para reemplazar a la sentencia `if` cuando el programa puede tomar múltiples rutas de ejecución.

- La **expresión** podrá tomar valores enteros o de tipo carácter.
- La expresión SIEMPRE debe ir entre paréntesis.
- **Funcionamiento**:
  - Se evalúa la expresión y se compara el valor con aquellos que van después de la palabra **`case`**.
  - Si alguno coincide, se ejecutan todas las instrucciones que vayan después de los dos puntos hasta encontrar un **`break`**.
  - No es obligatorio que haya un **`break`** por cada **`case`**, si no existe se seguirán ejecutando las instrucciones de los siguientes **`case`** hasta encontrar el **`break`** o hasta el final del **`switch`**.
  - Si no coincide ningún valor se ejecutan las instrucciones de la opción **`default`**.
- La opción **`default`** no es obligatoria.

1.3. Ejemplos: Sentencia `switch`

Lassiguientesdosexpresionescondicionalessonequivalentes:

```
if(a==1)
   printf("Es un uno");
else if(a==2)
   printf("Es un dos");
else
   printf("No es uno ni dos");
```

```
switch(a) {
   case 1: printf("Es un uno");
           break;
   case 2: printf("Es un dos");
           break;
   default: printf("No es uno ni dos");
}
```

Lavariable a es de tipo entero

Lassiguientesdosexpresionescondicionalessonequivalentes:

```
if(a==0 || a==1 || a==2 || a==3 || a==4)
   printf("a está en el rango 0-4");
```

```
switch(a) {
   case 0: case 1: case 2: case 3: case 4:
      printf("a está en el rango 0-4");
}
```


1.3. Ejemplo: Sentencia `switch`

Las siguientes dos expresiones condicionales son equivalentes:

```
if(operador=='+')
   resultado=a+b;
else if(operador=='-')
   resultado=a-b;
else if(operador=='*')
   resultado=a*b;
else if(operador=='/')
   resultado=a/b;
else puts("Operador inválido");
```

Lavariable operador es de tipo carácter

```
switch(operador) {
   case '+': resultado=a+b; break;
   case '-': resultado=a-b; break;
   case '*': resultado=a*b; break;
   case '/': resultado=a/b; break;
   default: puts("Operador inválido");
}
```


1.3. Ejemplo: Sentencia `switch`

```
#include <stdio.h>
void main() {
   int opcion;
   printf(" 1> España\n 2> Francia\n 3>
          Italia\n 4> Inglaterra\n");
   printf("Opción: ");
   scanf("%d",&opcion);
   switch (opcion) {
      case 1: printf("Hola\n"); break;
      case 2: printf("Allo\n"); break;
      case 3: printf("Pronto\n"); break;
      case 4: printf("Hello\n"); break;
   }
}
```


1.3. Ejemplo: Sentencia `switch`

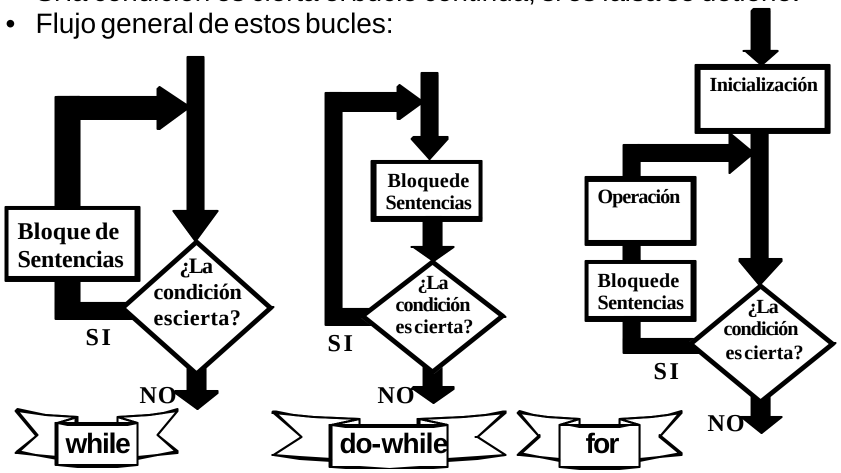
```
#include <stdio.h>
void main() {
   char letra;
   printf("Introduce una letra: ");
   scanf("%c",&letra);
   switch (letra) { /* Aquí la expresión es una variable de
      case 'a':      tipo carácter*/
      case 'e':
      case 'i':
      case 'o':
      case 'u': printf("Es una vocal minúscula\n");
                break;
      case 'A': case 'E': case 'I': case 'O':
      case 'U': printf("Es una vocal mayúscula\n");
                break;
      default: if (((letra >'a') && (letra <='z')) ||
                   (letra=='ñ'))
                printf("Consonante minúscula\n");
               else  printf("Consonante mayúscula\n");
   }
}
```

1.3. Ejemplo: Sentencia `switch`

```
/*Programa que calcula la
resistencia de un conductor
cilíndrico. R=ρxl/A, donde ρ es la
resistividad del conductor, l es
la longitud del conductor y A es
el área de la sección del
conductor*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <stdlib.h>
/*Definir las resistividades */
#define RHO_COBRE      17e-9
#define RHO_AL         25.4e-9
#define RHO_PLATA      16e-9
#define RHO_MANG       1400e-9
#define PI             3.14
void main() {
  float radio,l,area,rho,res;
  char ch;
  puts("Tipo de conductor: \
(C)obre, (A)luminio, (P)lata, \
(M)anganeso\n");
  ch= getchar(); /* Aceptar el tipo
                  de conductor */
  puts("Dame el radio y la
    longitud del conductor: ");
  scanf("%f %f", &radio, &l);
  area=PI*radio*radio;/* area del
                         conductor*/
  /* convertir a mayúsculas y
  determinar la resistividad */
  switch(toupper(ch)) {
    case 'C':rho=RHO_COBRE;
             break;
    case 'A':rho=RHO_AL; break;
    case 'P':rho=RHO_PLATA;
             break;
    case 'M': rho=RHO_MANG; break;
    default : puts("Opción no \
       válida"); exit(0); break;
  }
  res=rho*lon/area;
  printf("La resistencia del \
conductor es %.3e ohm", res);
}
```

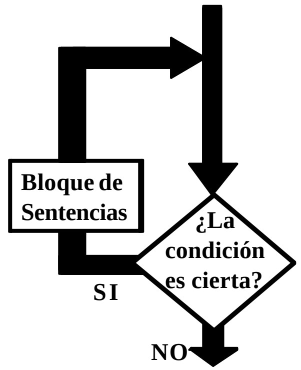

2. Estructuras Iterativas o Repetitivas

- Un proceso repetitivo o iterativo (bucle) permite que un conjunto de sentencias se ejecute varias veces.
- Hay tres tipos de bucles en C: **while, do-while** y **for**.
- Todos requieren una condición que determina si el bucle continua o no.
- Si la condición es cierta el bucle continua, si es falsa se detiene.
- Flujo general de estos bucles:



2.1. Bucle `while`

```
while (<condicion>) {
     Bloque de sentencias
}
```



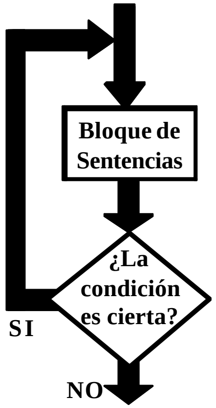
- **Funcionamiento**:
  - Se evalúa la condición de control del bucle.
  - Si la condición es cierta se ejecuta el Bloque de sentencias y se vuelve a evaluar la condición.
  - Si la condición es falsa se termina el bucle y se ejecuta lo que haya a continuación.
  - Es decir, **el bucle se ejecuta mientras la condición es cierta**.
- Importante: La condición de terminación se comprueba cada vez ANTES de ejecutarse el cuerpo del bucle (Bloque de Sentencias).
- Si la condición es inicialmente cierta, el bucle no terminará (bucle infinito) a menos que en el cuerpo del mismo se modifique de alguna forma la condición de control del bucle.

2.1. Ejemplo: Bucle `while`

```
/* Obtener la media de una lista de números. La
lista termina cuando se da el número cero */
#include <stdio.h>
void main() {
   int i=0;
   float x=0,media;
   printf("Dame un número: ");
   scanf("%f", &x);
   while(x!=0) {
      media=media+x;
      i++;
      printf("Dame otro número: ");
      scanf("%f", &x);
   }
   if(i!=0)
     printf("La media es %f.\n", media/i);
   else printf("No hay media.\n");
}
```

2.2. Bucle `do-while`

```
do {
  Bloque de sentencias
}  while (<condicion>)
```



- **Funcionamiento**:
  - Se ejecuta el Bloque de sentencias y luego se evalúa la condición.
  - Si ésta es cierta se vuelve al principio del bucle.
  - Si es falsa se termina.
- La condición se comprueba cada vez DESPUÉS de la ejecución del cuerpo del bucle.
- Por tanto, el Bloque de Sentencias siempre se ejecuta como mínimo una vez.

2.2. Ejemplo: Bucle `do-while`

```
/* Calcular el número más grande de una lista de
números mayores que cero. La entrada de números
terminará cuando se introduzca un número negativo
o cero. */
#include <stdio.h>
void main() {
   int num, max=0;
   do {
      printf("Dame un número: ");
      scanf("%d", &num);
      if(num>max)
         max=num;
   } while(num>0);
   if(max!=0)
     printf("El número más grande es %d.\n", max);
   else printf("No hay máximo.\n");
}
```

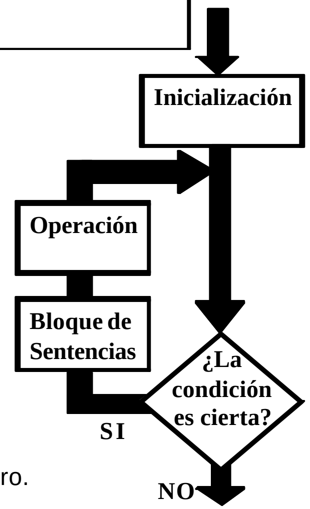

2.2. Ejemplo: Bucle `do-while`

```
/* Leer un valor que esté entre 0 y 100 */
#include <stdio.h>
#define VERDAD 1
#define FALSO 0
void main() {
   int num, ok=VERDAD;
   int retorno;
   do {
      printf("Dame un número: ");
      retorno=scanf("%d", &num);
      if((retorno!=1) || (num<0) || (num>100)){
         printf("Entrada no válida. ");
         ok=FALSO;
      }
   } while(!ok);
   printf("El número escrito es %d\n", num);
}
```

La función `scanf` devuelve el número de valores asignados. Puede ser 0 si hay algún error en la entrada. P. ej., si se lee un carácter para una asignación de tipo %d.

2.3. Bucle `for`

```
for(<inicialización>; <condición>; <operación>)
 {
   Bloque de Sentencias;
}
```



- **Funcionamiento**:
  - Se ejecutan todas las instrucciones de la inicialización.
  - Se evalúa la condición. Si es cierta se ejecuta el Bloque de sentencias y luego las instrucciones de operación en ese orden y se vuelve a evaluar la condición.
  - Si la condición es falsa termina el bucle.

NOTAS:

- En la parte de inicialización y de operación puede haber más de una instrucción, separadas por el operador coma (`,`).
- En la parte de condición sólo puede haber una.
- Si no hay condición, el test de condición es verdadero.
- Ninguna de las tres partes es obligatoria.

2.3. Ejemplos: Bucle `for`

(a) **`for(i=1;i<10;i++)`** ⇨ `i` empieza tomando el valor 1, cada vez que el bucle da una vuelta se incrementa en 1 (`i++`). El bucle parará cuando `i` sea igual a 10, es decir, el último valor que `i` tomará dentro del bucle será 9.

(b) **`for(i=100;i<500;i+=50)`** ⇨ `i` empieza tomando el valor 100; cada vez que el bucle da una vuelta se le suma 50 (`i+=50` es equivalente a `i=i+50`). Dentro del bucle `i` tomará los valores 100, 150, 200, 250,..., 400, 450.

(c) **`for( ; ; )`** ⇨ representa un bucle infinito!!

(d) **`for(i=2;i<=128;i*=2)`** ⇨ empieza con `i` igual a 2; cada vez que el bucle da una vuelta `i` se multiplica por 2. Esto continua mientras `i` sea menor o igual que 128. Los valores de `i` dentro del bucle serán 2,4,8,16,32,64 y 128.

(e) **`for(k=-20.2;h>32;z--,j++)`** ⇨ empieza con `k` igual a -20.2; el bucle se repetirá mientras `h` sea mayor que 32. Al final de cada bucle `z` se decrementa en 1 y `j` se incrementa en 1. La coma permite varias expresiones en el primer y tercer campo del `for`.

2.3. Ejemplo: Bucle `for`

```
/* Imprime los caracteres ASCII entre un valor
inicial y un valor final */
#include <stdio.h>
void main() {
  int i,ppio,fin;
  printf("Dame el valor inicial para los
      caracteres ASCII: ");
  scanf("%d", &ppio);
  printf("Dame el valor final para los
      caracteres ASCII: ");
  scanf("%d", &fin);
  puts("ENTERO  HEX  ASCII");
  for(i=ppio;i<=fin;i++)
    printf("%5d %5x  %5c\n",i,i,i);
}
```


2.3. Ejemplo: Bucle `for`

```
/* Generar la tabla de verdad de una función
digital */
#include <stdio.h>
#define FALSO 0
#define VERDAD 1
void main() {
  int A,B,C,Z;
  puts("Función lógica Z=((A AND B) NOR C)");
  puts("   A    B    C    Z");
  for(A=FALSO;A<=VERDAD;A++)
    for(B=FALSO;B<=VERDAD;B++)
      for(C=FALSO;C<=VERDAD;C++) {
        Z=!((A&&B) || C);
        printf("%4d %4d %4d %4d\n", A,B,C,Z);
      }
}
```


2.3. Ejemplo: Bucle `for`

```
/* Encontrar el primer número perfecto mayor que
28. Un número es perfecto si es igual a la suma de
sus divisores. Ej 6=1+2+3 */
#include <stdio.h>
void main() {
  int encontrado=0, intento,cont,suma;
  intento=29;
  while(!encontrado) {
    for(suma=1,cont=2;cont<intento;cont++)
      if(intento%cont==0) suma+=cont;
    if(suma==intento) entontrado=1;
    else intento++;
  }
  printf("Número perfecto mayor que 28 = %d.\n",
         intento);
}
```

Existe una forma más óptima de resolver este problema, encuéntrala.

2.4. *Sentencia* **break**

- La sentencia **`break`** se usa para salir de un bucle.
- Se puede usar con cualquiera de los tres bucles que hemos visto.
- Cuando se utiliza un bucle infinito (`while(1), do..while(TRUE), for(;;)`), sólo se puede salir de ellos usando la sentencia `break`.
- Ejemplo:

```
/* Escribe el cubo de una lista de números hasta que lee un 0 */
#include <stdio.h>
void main() {
  int i;
  for(;;) {
    puts("Dame un número para calcular el cubo");
    scanf("%d", &i);
    if(i==0) break; /* para el bucle infinito */
    printf("El cubo de %d es %d\n", i, i*i*i);
  }
  puts("Has dado un cero");
}
```

- Este programa puede ser escrito sin usar un bucle infinito. De hecho, no es recomendable usar bucles de este tipo.

2.5. *Sentencia* **continue**

- La sentencia **`continue`** se usa SÓLO dentro de un bucle.
- Termina la iteración que se esté ejecutando y se vuelve a evaluar la condición del `while`, del `do-while` o bien del `for`.
- Ejemplo: Muestra los valores desde 0 hasta 9 excepto el 5.

```
for(i=0;i<10;i++) {
    if(i==5) continue;
    printf("El valor de i es %d\n", i);
 }
```